(12) UK Patent Application (19) GB (11) 2 345 229 (13) A

(43) Date of A Publication 28.06.2000

(21) Application No 9828538.0

(22) Date of Filing 23.12.1998

(51) INT CL[7]
H04L 9/06

(52) UK CL (Edition R)
H4P PDCSP
U1S S2209

(56) Documents Cited
GB 2327581 A WO 97/22192 A1

(58) Field of Search
UK CL (Edition Q) H4P PDCSA PDCSP PDCSX
INT CL[6] H04L 9/06
Online:WPI,EPODOC,JAPIO,INSPEC

(71) Applicant(s)
Motorola Limited
(Incorporated in the United Kingdom)
Jays Close, Viables Industrial Estate, BASINGSTOKE, Hampshire, RG22 4PD, United Kingdom

(72) Inventor(s)
Bo Lin
Stephen McSpadden

(74) Agent and/or Address for Service
Sarah Gibson
Motorola Limited, European Intellectual Property Department, Midpoint,Alencon Link, BASINGSTOKE, Hampshire, RG21 7PL, United Kingdom

(54) Abstract Title
Encryption system resists differential power analysis attacks

(57) A message is encrypted using a block-cipher algorithm to protect again Differential Power Analysis (DPA) attacks. Blocks of the message are combined with blocks of the key using the algorithm, but the order in which the blocks are combined is made to vary for each round or iteration of the encryption operation. Preferably, the combination order is chosen randomly for each round, and randomization of the order is achieved either using a physical random number generator (RNG) or a pseudo RNG using part of the encryption key as a seed value. For added resistance to DPA attacks, the encryption process can include "dummy" computations periodically through the encryption sequence to throw-off analysis of power consumption data.

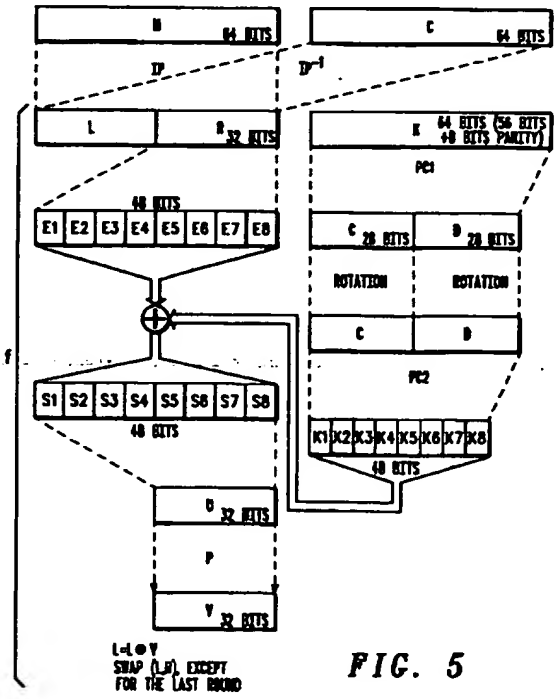

FIG. 5

GB 2 345 229 A

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

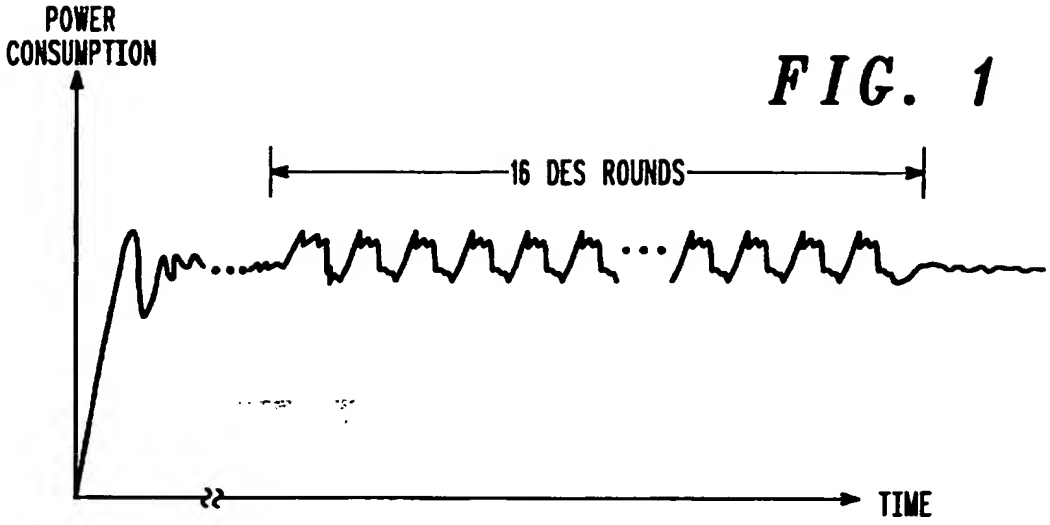

FIG. 1

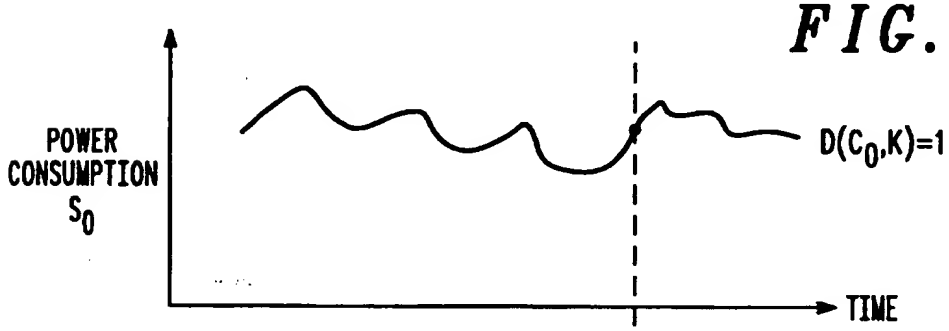

FIG. 2

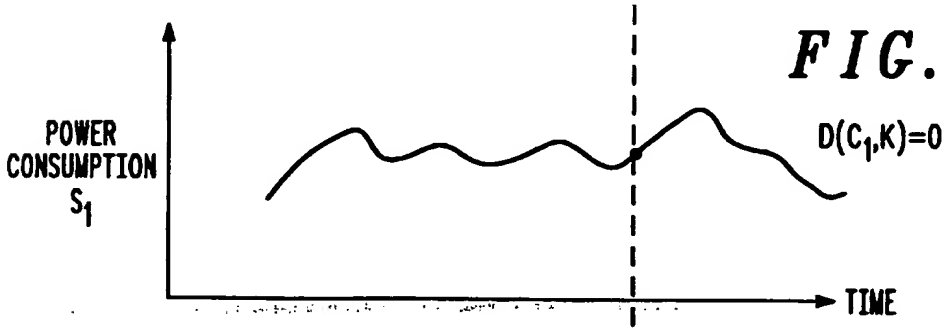

FIG. 3

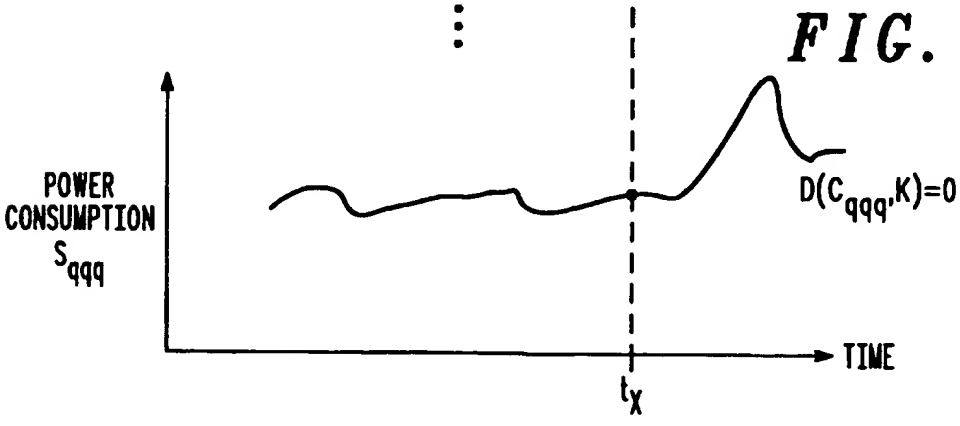

FIG. 4

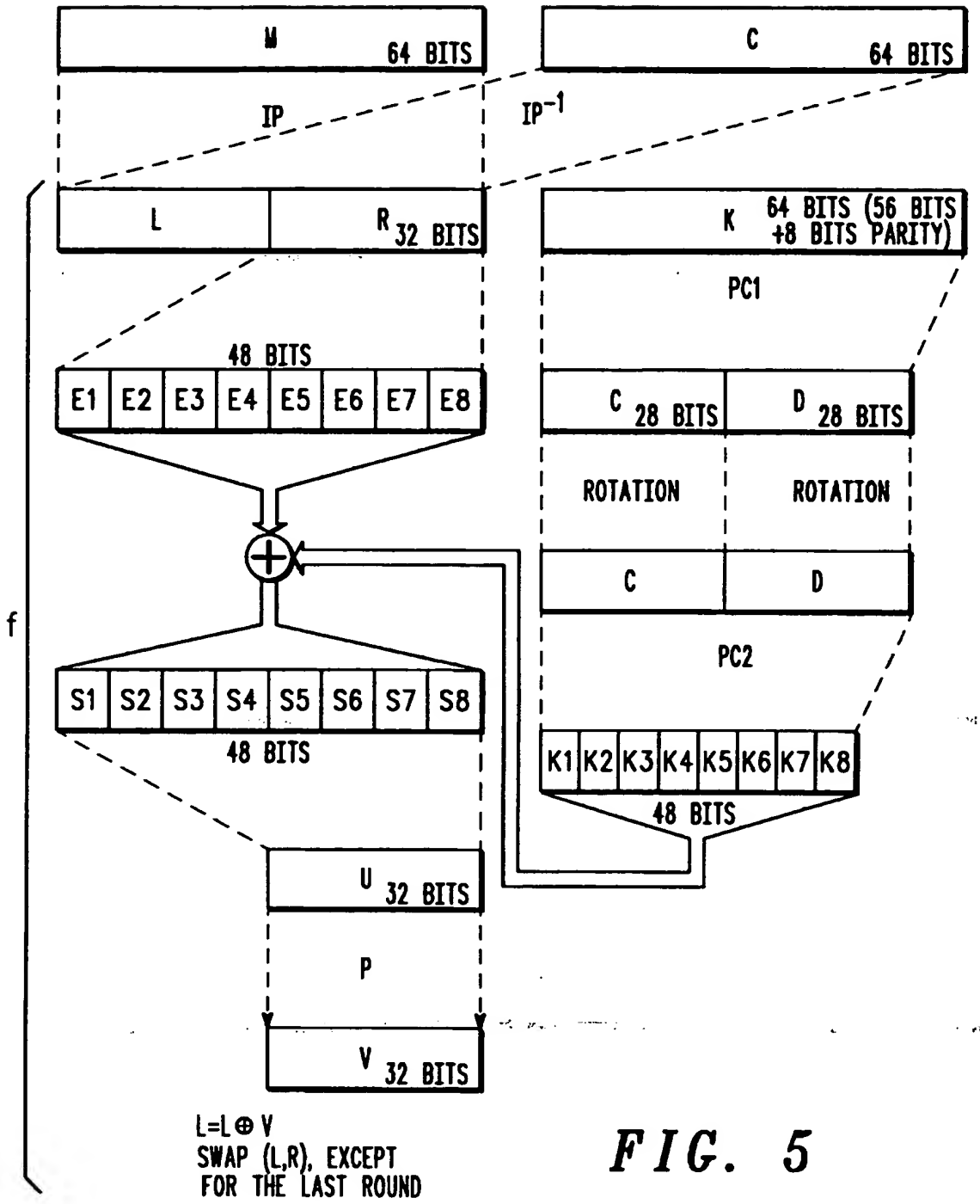
M
64 BITS
C
64 BITS
IP
IP-1
L
R
32 BITS
K
64 BITS (56 BITS +8 BITS PARITY)
PC1
48 BITS
E1 E2 E3 E4 E5 E6 E7 E8
C
28 BITS
D
28 BITS
ROTATION
ROTATION
C
D
PC2
f
S1 S2 S3 S4 S5 S6 S7 S8
48 BITS
K1 K2 K3 K4 K5 K6 K7 K8
48 BITS
U
32 BITS
P
V
32 BITS
L=L⊕V
SWAP (L,R), EXCEPT
FOR THE LAST ROUND

FIG. 5

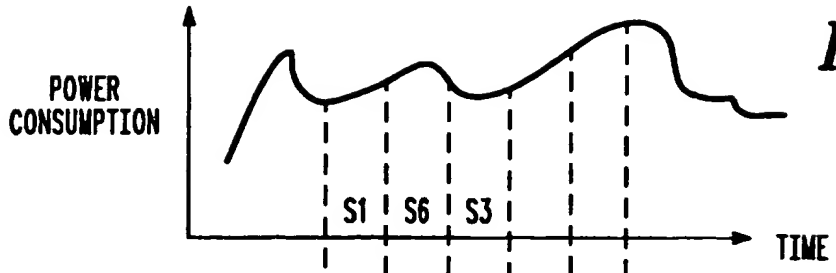

FIG. 6

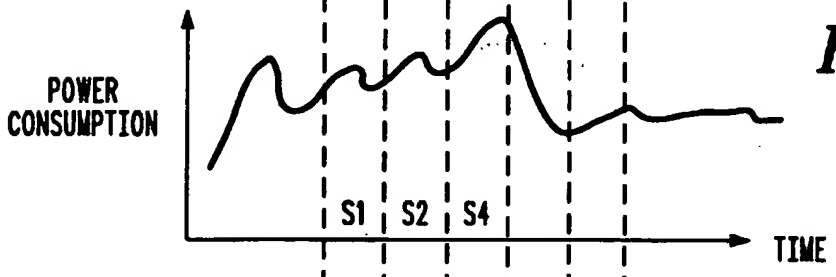

FIG. 7

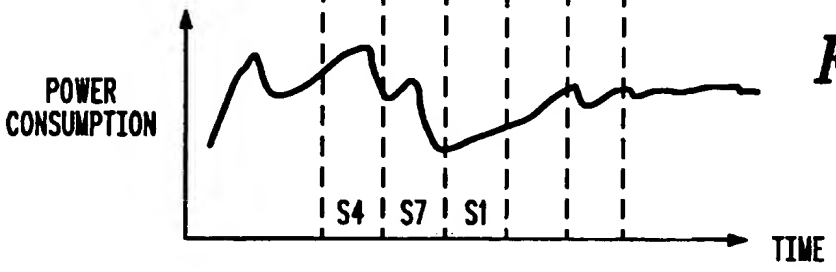

FIG. 8

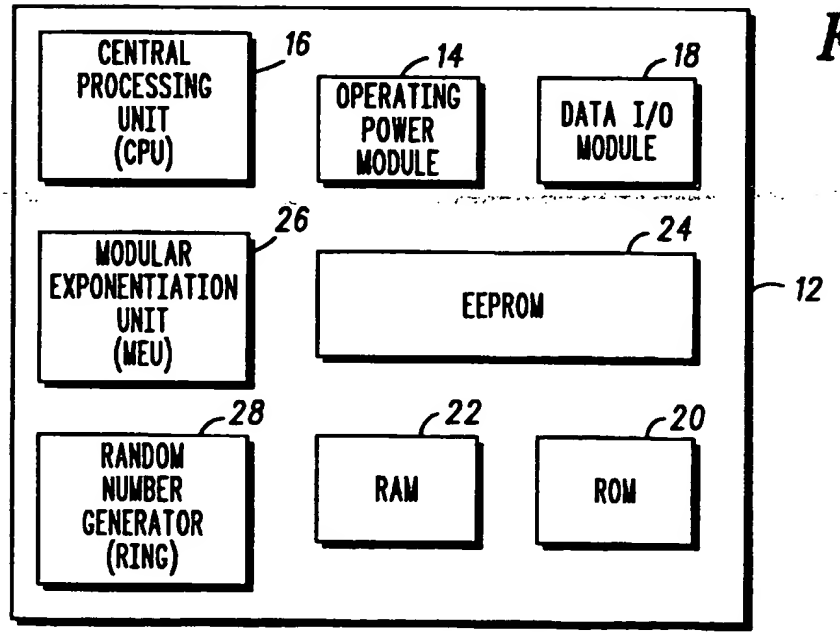

FIG. 9

2345229

METHOD FOR ENCRYPTING DATA

Field of the Invention

The invention relates to encryption and decryption of data, and more particularly to a method for encrypting or decrypting data with improved resistance to differential power analysis (DPA) attacks.

Background of the Invention

The use of data encryption is becoming an essential aspect of many electronic data transfer applications (e.g. Internet commerce, wireless telephony, banking, pay television, etc) as a means to prevent fraud. But as the use of encryption increases, so too does the effort which attackers take to break the keys used to encrypt the data because it becomes more lucrative to do so. Part of the increase in effort by attackers is the use of more sophisticated hacking techniques. With each new technique, industry must respond with more secure solutions. The threat of attackers is an issue so important that legitimate companies have been formed to try to break keys, and report how secure or insecure certain products are. Industry must therefore not only outsmart the illegal attackers, but the legitimate experts as well.

A hacking technique recently brought to the attention of the smartcard industry by one such group of experts is called a Different Power Analysis (DPA) attack on data which is encrypted using the Data Encryption Standard (DES). DES is a widely used, publicly available cryptography algorithm sanctioned by the Federal Information Processing Standard (FIPS 46-2) of the United States of America. DES is used to both encrypt and decrypt data, i.e. it is a symmetric algorithm. Although the algorithm is publicly known, use of DES nonetheless provides security because the key used to perform the encryption is not public and the algorithm is used multiple times using a new key each round to statistically make it difficult to decipher the resulting ciphertext message.

Generally speaking, a DPA attack is one in which an attacker monitors the power being consumed by the data processor performing DES and using statistical analysis and error correction is able to determine the key being used for encryption. In contrast, Simple Power Analysis (SPA) is a technique in which the data values can be read directly, without statistical analysis. After using DPA, to determine the key, the attacker can then decipher the data to determine the original message by using the DES algorithm.

A slightly more detailed explanation of DPA relies upon FIGs. 1-4. FIG. 1 shows a graph of power consumption of the processor performing the encryption as a function of time. An attacker can monitor power consumption, for example, using an oscilloscope and simply current probing the power pin, or by measuring the voltage across a small resistive load between the ground pin of the device under attack and the ground pin of the terminal. An example of what a portion of the output might look like is shown in FIG. 1. From the graph, it is apparent when a DES operation is taking place by observing 16 equal time periods of roughly the same power consumption pattern corresponding to the 16 rounds required in the DES algorithm. After determining when a DES operation begins, an attacker can then simply perform a large number of DES operations (e.g. 1000+) for random messages and sample the power consumption over one or more particular rounds of each DES operation, preferably over the last round. As FIGs. 2-4 illustrate, the attacker would then have 1000+ sample power consumption graphs (S_0, S_1, ... S_{999} +) corresponding to the same round or rounds within the DES operation. The attacker would likewise have 1000+ ciphertexts (C_0, C_1, ... C_{999} +) as the result of the 1000+ DES operations. These samples are obtained using the true encryption key, K, which is unknown to the attacker.

Using the data from the samples, the attacker then begins to try to guess the true key. In each round, DES uses a 48-bit binary key, divided into eight 6-bit sub-keys (K1-K8). So the attacker would use DES to decipher the encrypted message, i.e. the ciphertext, C, using the 2^6 (64) possible combinations for each of the 8 sub-keys. To figure out which of the guessed sub-keys is the real one, the attacker uses statistical analysis. For each guess, a D value is determined

for each message. More particularly, if a target bit of an input message after a DES initial permutation (IP) is equal to the corresponding target bit of deciphered data (i.e. after the P permutation), a function D(Ki, C) is set to be equal to 1, else function D(Ki, C) is equal to 0. A differential averaging function is then performed using the results of the function D and the data from the sample power graphs. For example, apply D-1/2 as a weight to every sample power graph, and sum all waveforms for each guessed key. If the guess is incorrect, the weighted sum of all waveforms will be close to 0, whereas the weighted sum for the true sub-key will be either much greater than 0 or much less than 0 (i.e. the weighted sum of the correct sub-key will be biased). This bias is sometimes referred to as the power consumption "spike" associated with the correct sub-key. This biasing effect is due to the following. The number of 0s and the number of 1s will be roughly balanced for any given message, based on probabilities. However, a 1 has more chance to correlate to higher (or lower) power consumption while a 0 has more chance to correlate to lower (or higher) power consumption due to the electrical characteristics of the processor performing the encryption. As an example, say there are 50 1s and 50 0s in a 100-bit message. The number of 1s and 0s is balanced, but 30 of the 50 1s correlate to lower power consumption and 30 of the 50 0s correlate to higher power consumption. DPA exploits this bias to find a correct key by calculate the difference in power consumption.

More detailed information on DPA attacks can be found, for example, at www.cryptogrpahy.com under the heading "DPA Technical Information."

One proposed solution to protect against DPA attacks is to introduce a random delay, in other words, to program the executing processor to insert a delay between each DES round, the length of which varies randomly. Thus, in reference to FIG. 1, rather than seeing 16 evenly spaced power consumption patterns, one would see 16 unevenly spaced patterns. However, this solution merely adds one more relatively easy step to an attacker's analysis, namely removing the random delay using SPA.

In view of the susceptibility of most data processing systems, particularly integrated circuits used in contact smartcards, to attacks using DPA techniques, card issuers, internet retailers, and consumers at large are demanding that safe guards be put in place to prevent deciphering of encrypted messages in this way. While certain commercially available encryption products do show the ability to withstand SPA attacks, Cryptography Research of San Francisco claims that no commercially available products have been found which resist DPA attacks. Accordingly, there is a need to protect encryption and decryption methods from this type attack, while at the same time ensuring an equivalent degree of ciphering.

Summary of the Invention

The present invention fulfills the above stated need in an encryption method that utilizes a block-cipher type of algorithm. In the method, a series of rounds are executed, wherein the order in which sub-keys and sub-blocks are combined in the algorithm is changed for each round of the method, preferably in a random order. By randomizing the order of combination, an attacker cannot simply guess a key and work backward from ciphertext to decipher a message. The attacker must further accurately guess the random order of combination. The invention is more fully defined in claim 1, and is more fully described in the following description taken in conjunction with the accompanying drawings.

Brief Description of the Drawings

FIG. 1 is a graph which plots power consumption of a data processor performing data encryption using DES as a function of time. This graph illustrates that 16 substantially similar time periods correspond to the 16 rounds called for in DES. Thus, an attacker is able to determine when an encryption/decryption operation is occurring.

FIGs. 2-4 illustrate a series of sample power consumption graphs, again plotted as a function of time, as one might obtain if trying to perform DPA. Note

that the scales of these figures are not necessarily the same as used in other graphs.

FIG. 5 is a flow diagram presenting how a message (M) is encrypted into ciphertext (C) in accordance with DES.

FIGs. 6-8 illustrate a series of sample power consumption graphs, as would be obtained by an attacker if trying to perform DPA on a device which uses an encryption method in accordance with the present invention. Again, the scales of these graphs do not necessarily correlate to the others.

Detailed Description of a Preferred Embodiment

For a fuller appreciation of the benefits and operation of the present invention, a slightly more detailed explanation of the encryption algorithm used would be advantageous. While the following discussion will focus on using the DES algorithm due to its widespread use and understanding, it is noted that the present invention can be implemented using any type of block-cipher algorithm, that is any algorithm in which portions or blocks of a message are combined with portions of a key to produce the encrypted message or ciphertext. An even more in depth explanation of DES can be found at the FIPS website for the standard, at www.dice.ucl.ac.be/crypto/standards/fips/html/fip46-2.htm.

As represented in FIG. 5, DES takes a 64-bit message (M) and converts it into 64-bits of ciphertext (C). This is achieved through a series of computations as follows. The input message is subjected to an initial permutation (IP) whereby the 64-bits are re-arranged into an order defined by DES. This permuted block is then input into a complex key-dependent computation called the cipher function *f*. The cipher function, more fully described below, is performed 16 times, each iteration being referred to as a "round." At the end of the 16 rounds, the cipher function result (called the pre-output) is then subjected to another permutation, which is the inverse of the initial permutation (IP^{-1}). The result of (IP^{-1}) is the finally encrypted message, i.e. the ciphertext, of the original message.

The cipher function takes a 32-bit portion of the input message, shown in FIG. 5 as a right portion, R, and performs an expansion function, E, to expand the block to 48-bits. The expansion function is dictated by the E bit selection table in the standard. The expanded input portion is then XORed with the key for the given round, in 6-bit portions. In other words, the 48-bit expanded input portion is segmented into eight 6-bit portions, E1-E8. Likewise, the 48-bit key for the round is segmented into eight 6-bit sub-keys K1-K8. A 6-bit sub-key is XORed with a corresponding portion of the expanded input, e.g. E1 is XORed with K1, E2 is XORed with K2, etc. The manner in which a key is derived for the input of the XOR, as shown in the right half of FIG. 5, is described more fully below. The result of the XOR produces eight 6-bit selection portions (S1-S8). Each of these selection portions is then subjected to a selection function, S. The selection function takes a 6-bit input (S1-S8 resulting from the XOR operation) and converts it into a 4-bit output. Again, the conversion is dictated by the selection tables in the standard. Each of the 8 6-bit inputs to the selection function is converted using its own selection table. In other words, there is a different table for each of portions S1-S8. The eight 4-bit outputs from the selection function form the 32-bit block U in FIG. 5. U is then subjected to a primitive permutation function, P, to produce a 32-bit block V. Again, the permutation function, P, is dictated by the table in the standard. Once V is obtained, V is XORed with L, the 32-bit left portion of the input message M. L is then swapped with R, and the process is repeated for the next round unless it is the last round (i.e. the 16^{th} round). On the last round, rather than swapping L and R, L and R as a 64-bit block undergoes the inverse of the initial permutation (IP^{-1}) to produce the final ciphertext C.

As mentioned above, a different key, K, is generated for each round of the cipher function, *f*. This key, called the round-key, is derived from the single key used for each DES operation. In other words, a single key will be used to encrypt and decrypt a message, but for each round of DES this single key is manipulated differently. In DES, each key is initially 64-bits, but one bit in each of the eight 8-bit bytes is used for error detection in key generation, distribution, and storage. More specifically, bits 8, 16, ..., 64 are used to assure that each byte of the key is of odd parity. In each round, the key, K, is subjected to a first

permuted choice function, PC1, dictated by the tables in the standard, to produce two different key portions, C and D. The parity bits of the original key are not used in the permuted choice function, thus C and D will each be a block of 28-bits. Blocks C and D then undergo a rotation as shown in FIG. 5. More specifically, the bits will undergo a left shift operation in the first round. In subsequent rounds, C and D initially begin as the C and D values of the previous round and then undergo either one or two left shift operations, depending upon which round it is. The number of left shifts for any given round is dictated by the standard. After the appropriate rotation for the given round, blocks C and D are subjected to a second permuted choice function, PC2, again as set forth in the PC2 table of the standard. The resulting key is a 48-bit key for that round (called a round-key) which is segmented into eight 6-bit sub-keys (K1-K8).

As stated earlier, an attacker of a smartcard or other data processing system which uses encryption can learn when DES rounds are occurring. Using statistical analysis of sampled power consumption graphs (also known as "power signatures"), the attacker can determine the key being used to encrypt the message, and thus decipher the message by working the DES algorithm in reverse using that key. There are several reasons why this is possible. One reason is that a distinguishing power signature exists which tips off an attacker as to when DES is occurring. But another reason has to due with the nature of the DES algorithm itself. First, it is publicly available. But the solution to the problem is not necessarily to use secret encryption algorithms because there are legitimate national security and public policy reasons why at least certain encryption algorithms should be made public. Another characteristic of DES making it susceptible to attack is that with every step or computation, an attacker is able to identify corresponding bit locations. For example, the standard dictates how the bits of the right portion of a message, R, are expanded from 32-bits to 48-bits, and how a 64-bit key is broken into two 28-bit portions, and how the selection function S converts a 6-bit input to a 4-bit output to produce U.

The present invention enables one to utilize DES and other block-cipher algorithms to encrypt and decrypt messages, but with the added advantage of a degree of randomness in the execution of the algorithm. The order in which

certain operations occur can be changed without changing the resulting output of each DES round. And by changing this order each round, an attacker is unable to make a direct correlation between even the largest number of sample power signatures.

The invention as it relates to DES concerns the order in which the expanded portions E1-E8 of the input message are combined with sub-keys K1-K8 and/or the order in which the selection function S is performed on the resulting combination. In prior art DES implementations, E1 would be XORed with K1 to produce a 6-bit selection function input, S1. This S1 value would then be subjected to the selection function, i.e. converted to a 4-bit value using the S1 look-up table given in the standard. Similarly, E2 would be XORed with K2 to produce S2, which would be converted to a 4-bit value using the S2 look-up table. In some implementations, all XOR functions between the expanded portions and sub-keys would be performed before the selection function look-ups (i.e. E1-E8 would be XORed with K1-K8, respectively 6-bits at a time, to produce S1-S8 and then S1-S8 would be converted to 8 4-bit values using the 8 separate look-up tables). In either implementation, however, the order of the XOR function and the selection function look-up were the same for every round.

In a preferred implementation of the present invention with DES, the order in which the XOR combination of Ei and Ki and the selection function look-ups are performed is changed from round to round, preferably in a random manner. Thus, if an attacker is analyzing power signatures, the power consumption being analyzed at a given point in time, or over a given time period, during the cipher function for each sample will be different. For example, in reference to FIGs. 6-8, what corresponds to the power being consumed during the XOR/look-up for S2 in Sample 1 (FIG. 6) is actually the power being consumed for the S1 XOR/look-up in Sample 2 (FIG. 7) and the S4 XOR/look-up in Sample 3 (FIG. 8). An attacker will either assume that the order is the same from sample to sample (and therefore will not be able to guess the key using the technique described above), or will have to analyze the sample graphs using all 8! (8 factorial) possible order combinations. Randomizing the order in which the S block look-up and the XOR are performed for each round and for each DES operation

provides a level of security above merely changing the order for each round in a predictable manner for each DES operation. For example, while the order of the S block XOR/look-up may be S14, S7, S1, S3, S9, S11, S2, S4, S14, S5, S15, S13, S12, S8, S10, S6 in one DES operation, it will be a complete different order for the next DES operation.

While preferably both the order of the XOR operation and the selection function look-up are randomized, it is noted that randomizing only one of these will improve the resistance to a DPA attack. For example, Ei and Ki can be XORed in a random order to produce the inputs to S1-S8 (although in the XOR, "i" is equal so that E1 is always XORed with K1, E2 is always XORed with K2, etc.), but then the selection function look-up occurs in sequential order, S1, S2, S3, S4, S5, S6, S7, S8. Similarly, Ei and Ki can be XORed sequentially, followed by a random selection function look-up. However, even better resistance to a DPA attack is achieved by randomizing both orders. In general, it is noted that one can randomize the order in which one or more functions occur for any function which involves a data value which is dependent on, or derived from, the key to improve the resistance to attack. However, the final encryption or decryption output should not be affected by the change in order. Another example of an order which can be changed to improve resistance to attack is the order in which the C and D values are rotated or shifted. DES dictates how many shifts should be performed depending on which round it is, but it does not matter whether C or D is shifted first. By randomly changing the order in which C and D are shifted, one can further protect against an attack.

The manner in which the function order is randomized, be it the XOR function, the selection function, the rotation of C and D, or the like, will likely be dependent upon the configuration of the particular data processing system being used. For example, if a physical random number generator (RNG) is available, as in FIG. 9, the physical RNG can select the random order. A physical RNG is an RNG that generates a random number based upon sample "noise" from reverse-biased diodes, oscillator phase noise, or other physical phenomena. A physical RNG is usually implemented in hardware, e.g. by a combination of two independent oscillator circuits which are cascaded together, whereby the current

source of the second oscillator is modulated by the saw-tooth wave-form output of the first oscillator to produce a pulse train representing a binary random number. Physical RNGs are sometimes used to generate encryption keys, so the integrated circuit to be used for encryption may already include a physical RNG which can likewise be used to randomize the S block look-up order.

If a physical RNG is not available, a pseudo RNG can be used. A pseudo RNG is one that generates a random number through algorithmic manipulation of a seed value using software. In accordance with a preferred embodiment of the present invention, the seed value is a hash value related to the key, K. Any standard hashing function is performed on a byte of the key, K. The result of the hashing function is a randomized portion of the key, which is then used as the seed value for any standard pseudo RNG algorithm. The result will be a pseudo random 8-bit (1 byte) number, r. This result, a pseudo random number dependent on the key, could be used to determine the order in which the function is performed. However, an even better option is further obfuscate the derivation of the random number by making it dependent on the message as well. In a preferred embodiment, this resulting random number, r, is XORed with a byte from L, the left portion of the input message for a given round. The result of this XOR -- r $\oplus$ [a byte of L]-- is then used to select the order in which the function is performed.

The above methods are merely examples of how one might choose a random order for the selection function. There are numerous randomization techniques available that could be used with the present invention, each with its own benefits and shortcomings. The selection criteria for a technique will likely be dependent on the degree of randomness the technique repeated provides, and the ease at which the technique can be implemented in the user's data processing system. Furthermore, the application in which the encryption device is to be used will likely determine the extent to which the device must be resistant to attack since types of message will contain more sensitive information than others. The invention is not intended to be limited to a specific method for choosing the random order, unless a claim expressly specifies such a method.

While randomizing the order in which the selection function look-up is performed in each round helps to strengthen the resistance to a DPA attack, a determined attacker with sufficient computing power can nonetheless break the key by analyzing the sample power graphs over all 8! (8 factorial) possible order combinations of the S blocks. Therefore additional measures can be taken if deemed necessary. For example, as stated above, one can randomize both the selection function look-up and the XOR function. By increasing the number of functions which are randomized, one makes it more difficult to decipher a key.

Another technique which could be used to improve resistance to attacks is to insert a "dummy" operation to confuse analysis of a power signature. For example, one could insert "dummy" S block look-ups into the DES routine, whereby an S block look-up is performed but the result or output of the look-up is not included in the pre-output value, U, but is instead written elsewhere and not used. Thus, the power signature will appear as if an S block look-up has occurred, but an attacker will not be able to determine if the output from the look-up is included in the pre-output value, U. The number of dummy look-ups performed can be chosen to optimize the time it takes to perform the DES operation and the benefit gained in DPA attack resistance. For example, one may choose to only insert one dummy look-up per round in the interest of execution time, but one or more dummy look-ups for each real look-up would make the DES operation more resistant to attack. The programmer must strike the appropriate balance for a given application. Preferably, one performs at least one dummy look-up (and more preferably 2-4 dummy look-ups) for each real look-up. In other words, if the randomly chosen order for the S block look-ups is S7, S1, S3, S2, S4, S5, S8, S6, in a round, then one might instead perform the look-ups as be S7, D1, D2, D3, S1, D4, D5, D6, S3, D7, D8, D9, where Di is a dummy look-up.

An exemplary implementation of the present invention can be understood in reference to FIG. 9. FIG. 9 illustrates, at a functional block level, a data processing system in the form of an integrated circuit (IC) 12 which is suitable for use in practicing the present invention. While IC 12 as shown is intended for use in a smart card, or personal data carrier, this is not a requirement of the present

invention. Nor is the invention limited to performance by an integrated circuit. Rather, the invention is useful in any application which employs use of an block-cipher encryption algorithm. It is also noted that while IC 12 is illustrated to include definitive blocks for performing particular functions, as further described below, in practice the particular functional blocks of IC 12 are likely to not be clearly identifiable as "blocks" on the actual manufactured IC. Furthermore, the arrangement of such "blocks" on the chip are likely to not correspond to the arrangement shown. It is also noted that while IC 12 may be illustrated in the form of a single semiconductor die, the functionality described can be implemented in one or more die. Moreover, an IC may include functional blocks other than those illustrated in FIG. 9. Accordingly, the figures are not intended to limit the scope or applicability of the invention unless expressly indicated otherwise.

In IC 12, an operating power module 14 receives operating power by either direct contact with contacts between a reader terminal and the smart card, or by radio frequency (RF) transmission from the reader terminal in the case of a contactless smart card. Power module 14 provides a positive power supply potential (e.g. V_{DD}) to the other circuitry in IC 12. A central processing unit (CPU) 16, also referred to a microprocessor or microcontroller core, performs the control, timing, and decision making functions. For example, CPU 16 controls the read, write and erase operations to the memory and makes data available to data I/O module 18. Data I/O module 18 sends and receives data to and from the reader terminal.

A Read-Only-Memory (ROM) module 20 of IC 12 stores the program instructions for the given card application which are set during the IC manufacturing process and then executed by CPU 16. For example, ROM may include the program instructions necessary for the CPU to perform DES or other encryption algorithm. ROM may also include "look-up" table values as may be called for in such algorithms. A Random-Access-Memory (RAM) module 22 is also included. RAM is volatile memory and thus provides temporary storage of information. Electrically Erasable Programmable ROM (EEPROM) 24 is a non-volatile memory array of IC 12 that stores the primary information of the card,

such as personal identification, medical history, banking information, monetary values, security codes, etc. depending on the card application. While such information is stored digitally, as binary numbers, the origination message is usually an alphanumeric which was been converted. It is this primary information which will need to be encrypted when transferring the data to a reader, and decrypted when transferring the data from a reader. While EEPROM is a preferred form of memory, other types of non-volatile memory can be used in place of EEPROM 24.

IC 12 further includes a Modular Exponentiation Unit (MEU) 26 which is used to encrypt data being transferred between the card and the reader and to decrypt data being transferred from the reader to the card. In accordance with the present invention, MEU is used to perform the operations of a block-cipher encryption algorithm, such as DES. The MEU, under the control of the CPU, manipulates data values stored in the various memory blocks to produce the encrypted/decrypted results and to store these results back in memory. The CPU then controls the use of the results.

IC 12 also includes a physical Random Number Generator (RNG) 28 suitable for use in conjunction with the present invention as described above. RNG 28 need not be of any particular circuit design for use with the present invention. Any known or to-be-developed physical RNG (as defined previously) can be used, or as stated above a pseudo RNG can be used to generate a random number which selects a particular function order for the encryption algorithm to protect against power analysis attacks as described above.

The foregoing description and illustrations contained herein demonstrate many of the features and advantages of the present invention. In particular, it has been revealed that by changing the execution order of a function(s) within a block-cipher algorithm, the susceptibility of successful DPA attack on the encrypted message is significantly reduced. Moreover, the resistance to such attacks is greatly improved if the order of function is selected randomly, and changes for each round or iteration of the encryption operation. As an example, an experiment showed that by using DPA techniques on a particular integrated

circuit which executes the DES encryption algorithm, the encryption key (i.e. the power consumption spike) could be discovered after processing and analyzing just 100 random messages. In contrast, the power consumption spike could not be detected even after processing and analyzing 2000 random messages when randomizing both the XOR function and the selection function look-up in conjunction with the same integrated circuit. And because use of the invention has no impact on the output of the encryption process, a legitimate receiver of the message will be able to decrypt the message as usual.

Thus it is apparent that there has been provided, in accordance with the invention, a method for encrypting a message using a data processing system that fully meets the need and advantages set forth previously. Although the invention has been described and illustrated with reference to specific embodiments thereof, it is not intended that the invention be limited to these illustrative embodiments. Those skilled in the art will recognize that modifications and variations can be made without departing from the invention. For example, the invention is not limited to DES. It can be implemented on any block-cipher encryption/decryption algorithm, whereby the message and key are combined on a block-by-block basis. In addition, the invention is not limited to the particular method by which the order of combination is varied (e.g. using hardware or a physical RNG or using software or a pseudo RNG). It is also important to note that the present invention is not limited in any way to a particular type of data processing system. While in its most common form the data processing system will likely be an integrated circuit which includes a microcontroller or microprocessor unit to perform the various algorithm functions, such is not a requirement for practicing the invention. Nor is the invention limited to any particular use or application of the resulting encrypted message. Therefore it is intended that the invention encompass all variations and modifications as fall within the scope of the appended claims.

Claims

1. A method for encrypting a message into ciphertext using a data processing system and a block-cipher algorithm, the method comprising multiple rounds, each round comprising the steps of:
   receiving a block of data, wherein the data is the message to be encrypted if the current round is the first round, else the data is from a previous round;
   deriving a round-key to be used for the current round;
   partitioning the round-key into several sub-keys;
   partitioning the block of data into several sub-blocks; and
   combining each sub-key with its corresponding sub-block using the block-cipher algorithm to generate a corresponding block for a next round, or to produce a portion of the ciphertext if the current round is the last round to be performed;
   characterised in that the order in which the sub-keys and sub-blocks are combined is changed for each round.

2. The method of claim 1 wherein the order in which the sub-keys and sub-blocks are combined is changed randomly for each round.

3. The method of claim 2 wherein the order is randomly chosen using a physical random number generator.

4. The method of claim 2 wherein the order is randomly chosen using a pseudo random number generator which uses a seed value derived from a key used to encrypt the message.

5. The method of any preceding claim further comprising the step of combining a sub-key with a sub-block to generate a dummy result, and wherein the dummy result is not included in the corresponding block for the next round or in the portion of the ciphertext if the current round is the last round to be performed.

6. The method of claim 5 wherein the step of combining a sub-key with a sub-block to generate a dummy result is performed more than once for each combination of a sub-key and sub-block which produces the corresponding block for the next round, or the portion of the ciphertext if the current round is the last round to be performed.

7. The method of any preceding claim wherein the cipher block algorithm is the Data Encryption Standard (DES) as set forth in the Federal Information Processing Standards Publication 46-2.

8. The method of claim 1 wherein the step of combining comprises XORing each sub-key with its corresponding sub-block, the result of XORing producing a corresponding selection block.

9. The method of claim 8 where the step of combining further comprises converting each selection block to another value based upon a corresponding selection function as stated in the Federal Information Processing Standards Publication 46-2.

10. The method of any preceding claim wherein the data processing system is an integrated circuit.

11. A method for encrypting a message into ciphertext using a data processing system and block-cipher algorithm as hereinbefore described with reference to the accompanying figures.

12. A data processing system for encrypting a message using a block-cipher algorithm comprising:

storage means for storing the message;

storage means for storing a key to be used for encrypting the message;

means for combining a portion of the message, or a value derived therefrom, with a portion of the key, or a value derived therefrom, to generate an encrypted block of data;

characterised as further comprising:

means for altering the order in which a portion of the message, or a value derived therefrom, is combined with a portion of the key, or a value derived therefrom, for each round of the block-cipher algorithm.

13. The data processing system of claim 12 wherein the means for altering comprises a physical random number generator.

14. The data processing system of claim 12 wherein the means for altering comprises a pseudo random number generator.

15. The data processing system of claim 13 or 14 as implemented in the form of an integrated circuit.

The Patent Office

18

INVESTOR IN PEOPLE

Application No: GB 9828538.0 **Examiner:** B.J.SPEAR
Claims searched: 1-15 **Date of search:** 1 September 1999

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.Q): H4P (PDCSA, PDCSP, PDCSX)

Int Cl (Ed.6): H04L 9/06

Other: Online:WPI, EPODOC, JAPIO, INSPEC

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
XE	GB2327581A	(Samsung) Whole document,eg Figs. 1,2a, p 7 l 29 to p 12 l 34 claims 1,2	1,7,8,12 at least
X	WO97/22192A1	(Northern Telecom) Whole document, eg Fig. 1 and p 3 l 12 to p 11 l 10	1,7,8 ,12 at least

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.